Adjusting Images for Different Sizes

In some circumstances, a graphic image might need to fit in a different amount of space than originally intended. For example, a user may adjust the global scaling constant using the Custom Font Size feature, or an application might allow the user to choose a differently sized font. In these cases, an application might draw an image that appears unexpectedly out of scale with the surrounding text or other window elements. Different methods of adjusting images are described in the following sections.

Alternatives to bitmaps

Adjusting images is especially a problem for bitmaps, which are not automatically scaled and do not usually look good when stretched. The best solution is to avoid using bitmaps altogether and use instead another kind of image that is designed to scale well. The following alternatives can be used in place of bitmaps:

- Metafiles are a convenient way to encapsulate an image for easy playback. They can automatically be scaled to fit the destination rectangle and normally look good at almost any size. (Of course, this benefit is lost if the metafile itself contains things that do not scale, such as bitmaps.)
- If the image is simple and you do not need to encapsulate it for easy storage and playback, you can create a routine that draws it on the fly using Windows graphic functions.
- TrueType glyphs can represent the image. This solution can be expensive if the expertise to create glyphs is not available in-house, but the image can be optimized for any size. A complex TrueType glyph will not be recognized by an accessibility aid, so you should label the graphic using the techniques described earlier in "Use of Bitmapped Text."

Accommodating bitmaps when changing size

Several methods can be used to accommodate a bitmap to differently sized regions:

- You can scale your bitmap "at run time" using the **StretchBlt** function so that it will be sized appropriately for its screen location.
- If you do not stretch your bitmap for an enlarged space, you can draw it at its normal size and leave blank space around it. However, to ensure that the region around your bitmap does not have unrelated, older information lying around, you should erase the region to the appropriate background color. You should specify the size of the region using a drawing method other than MM_TEXT so that it will automatically adjust to any global scaling factor.
- If you do not shrink your bitmap to fit a smaller space, you should make sure that the bitmap is properly clipped to the surrounding rectangle—for example, when the Custom Font Size feature is used to select a global scaling ratio of less than 100%. You should specify the size of the region using a drawing method other than MM_TEXT so that it will automatically adjust to any global scaling factor.

Sound

An application should not convey important information by sound alone, because some users will not be able to hear or recognize it. The user may be deaf or hard-of-hearing or may simply be using the computer in a very noisy environment, such as a workshop, or in a very quiet environment where it is inappropriate for the computer to be making sounds.

Applications make sounds for a variety of reasons. There are four different types of sound:

- Important sounds, which convey information that is not presented visually and which is important to the operation of the application. Examples include an audio wave file with narrative instructions or a sound notifying the user that new mail has arrived.
- Redundant alerts, which accompany a visual presentation of information, yet serve an additional purpose of attracting the attention of a user who is not looking directly at the computer screen. An example is the optional beep that can accompany a message box.
- Redundant sounds, which repeat information already presented visually and are not required for proper operation of the application.

An example is an error sound or beep that is heard when the user tries to move beyond the end of a list box.

- Decorative sounds, which enhance the appearance or presentation of an application, but are not required for its operation. Examples include the sound effects that accompany minimizing a window or activating a menu and background sounds and music used to establish a mood in many multimedia games.

The user who cannot hear redundant or decorative sounds is not disadvantaged, but the first two types of sound, important sounds and redundant alerts, do require special attention.

For redundant alerts, Windows 95 and other utilities have the capability to detect when the computer is making noise and to display a generic visual indicator to the user. This feature is referred to as "SoundSentry." This feature works reasonably well in cases where the sound is just a generic beep that is warning the user or trying to attract his or her attention. However, it is of limited use with applications that use different sounds to convey more complex information.

Conveying important sounds and complex information requires the cooperation of the application.

Supporting the ShowSounds Flag

To convey complex information to users who cannot rely on audible forms, the computer industry has standardized a concept called "ShowSounds." ShowSounds is a global flag that can be set by the user to indicate that he or she needs important information displayed by visual means. It requests applications to display the equivalent of closed captions for their sounds.

Applications can check the ShowSounds flag by calling the **SystemParametersInfo** function with the SPI_GETSHOWSOUNDS value.

Use of the ShowSounds flag does not mean that sounds cannot be presented normally. In fact, redundant use of sound and visuals generally increases the usability of an application. The user should be able to request visual feedback independently of whether they want audible feedback.

The ShowSounds flag is only applicable to applications that would normally present important information by sound alone. The application is responsible for determining how to convey the information in visual form. Examples in the following sections can help you determine behavior appropriate to your situation.

Audible alerts

To attract the user's attention, such as when new email arrives, an application might use the following techniques:

- Flash its title bar by using the **FlashWindow** function. If the window is not visible, the function will automatically flash the application's button on the taskbar.
- Display a message box that acquires the activation and keyboard focus. This technique should be avoided if the user might be typing into another application at the time.
- Display a status indicator on the notification area of the taskbar. This indicator should also flash when initially displayed to help attract the user's attention.

Redundant sounds

Applications often make a sound to indicate an error status—for example, when the user types an invalid character. In these cases, the application might flash its title bar by using the **FlashWindow** function. If the window is not visible, it will automatically flash the application's button on the taskbar.

Playing a video clip

Applications that display multimedia animation of video clips should support the ShowSounds flag with true closed captioning. Microsoft Video for Windows supports creating a separate, synchronized data stream for captioning information, although the application is responsible for displaying the information on the screen. Captioning is only necessary if the video clip has an audio track containing important information.

Playing an audio wave file

Applications that play audio wave files containing important information should display closed captions for that information. However, it may be difficult in some cases to synchronize the captions with the audio. For brief wave files, the captions can be displayed in a status bar or elsewhere in the application window, or in a floating window similar to a tool tip control. Longer descriptions can be displayed in a separate window. The user should be able to scroll through the information at his or her own pace. In hypertext applications, a link can be used to take the user to a separate screen containing the textual descriptions.

Turning Off Sounds

You should give the user the option to turn off sounds that your application makes, because sounds can be distracting or annoying for some people, such as those who are deaf or hard-of-hearing, or be inappropriate in some environments, such as crowded or public spaces. This is especially true of decorative sounds or sounds that are redundant to information on the screen.

If you do not want to provide your own option to turn off sounds, you can check the SM_BEEP option using the **GetSystemMetrics** function. If this option is FALSE, the user has chosen to turn off the standard system beep, and you can infer that they also want other sounds turned off as well.

Supporting System Sound Events

You should generate the proper system sounds for any action your application carries out. Windows 95 defines a large number of system sound events and plays the associated sound when an error occurs. When your application carries out actions that correspond to system events, it should use the **PlaySound** function to play the sound for the event if any has been set up in Control Panel. This makes your application consistent with the rest of the Windows environment and enables users to customize the sound scheme to give more or fewer sounds, as they prefer.

For more information about playing sounds, see, "Playing Sounds Specified in the Registry," in the Win32 Software Development Kit (SDK).

Defining Application-Specific Sound Events

You should allow the user to customize the way your application uses sounds by defining as many new sound events as possible. In Windows 95 and Windows NT, sound events are defined in the registry, and the user can use Control Panel to associate any sound, or no sound, with each event. You can have most of your events generate no sounds by default, but the user who desires additional audio feedback can use Control Panel to add their own sounds. This capability is especially useful for people with visual and some types of cognitive impairments.

For more information about playing sounds, see, "Playing Sounds Specified In the Registry," in the Win32 SDK.

Layout

There are several ways that the visual design, or layout, of an application can improve its accessibility.

Attaching Textual Labels to Controls and Graphic Objects

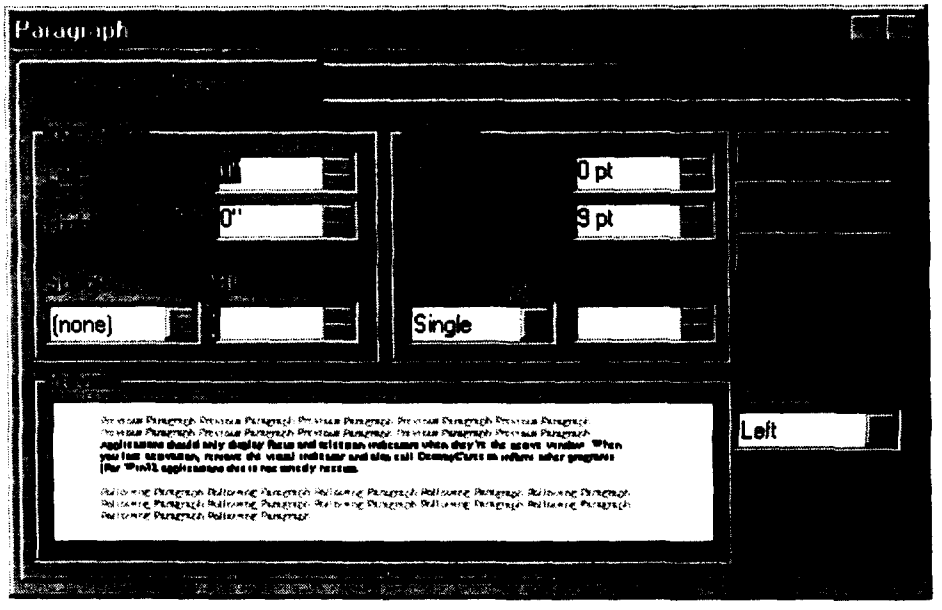
Some types of controls, such as buttons, have their own textual labels. Screen review utilities have no trouble describing such controls to the user, and voice input utilities can recognize the label name when it is spoken as a command. However, other objects, such as edit controls or graphical objects, are typically labeled by placing a static control nearby.

The Windows Interface Guidelines for Software Design describes guidelines for the placement of labels so that they are consistent across all applications. Proper labeling helps make the interface more consistent between applications and more usable for everyone; it also helps accessibility aids. Proper labeling allows a screen review utility to infer the relationship between a static control and the control associated with it. If a static control ends in a colon, the screen review utility knows it is a label and looks for an unlabeled control either to its right

or directly below it. When the utility describes the control to the user, it can use the label from the static control.

If a static control label and the control it refers to are not arranged in a standard pattern, it may be difficult for both sighted users and screen review utilities to determine which label applies to which control, or even that the two are related. A control and its label should not be separated by too great a distance, and a text label should not have unlabeled controls both beneath and to the right of it.

The following illustration shows the positioning of static control labels above and to the left of the unlabeled controls associated with them.

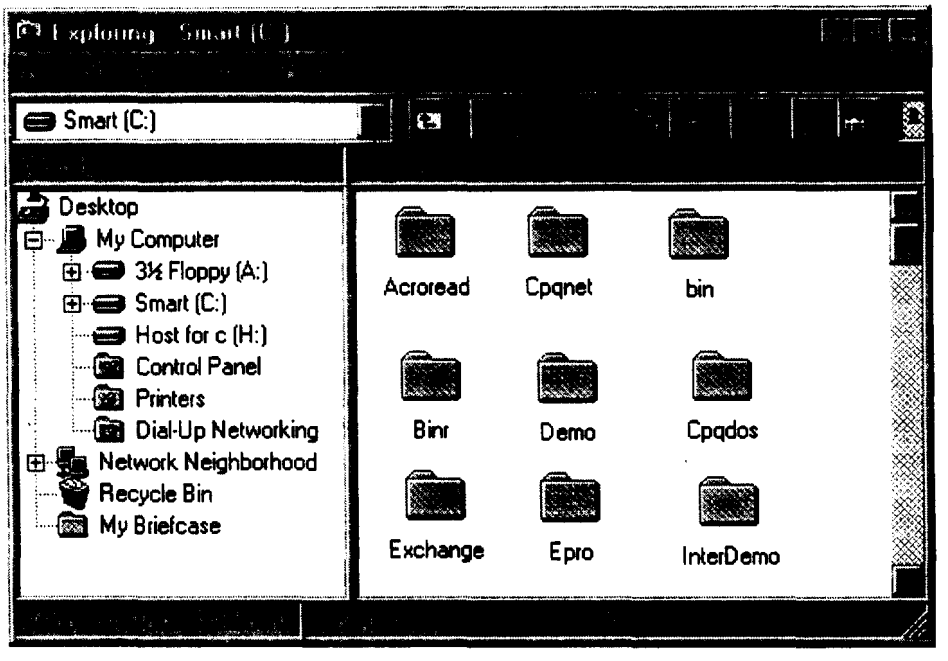


Labeling Icons

If your application uses an icon or any type of graphic to represent an object or control, you should also display a text label with it. This arrangement is already familiar to users, and the combination of text and graphic helps shorten the learning curve for a new user. It also helps screen review utilities describe the object to a user who is blind and helps users associate a name with the control to activate or navigate to it by voice or other means.

Following the normal guidelines, a text label should generally be placed immediately beneath a large icon or to the right of a small icon. When you place a text label beneath an icon, use the font, size, and color defined for icon titles in Control Panel to make it consistent with other applications. If you cannot display the text label visibly, at least make the label available in a tool tip control, as described previously.

The following illustration shows the proper positioning of text labels for both large and small icons.



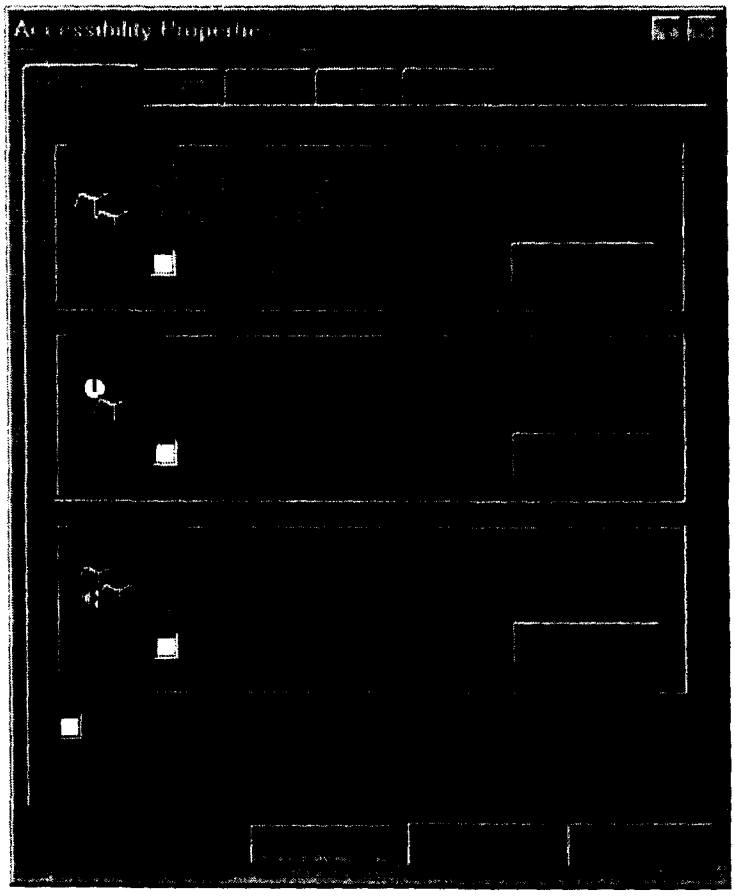
Labeling Controls Clearly

You should label controls and similar objects with names that convey information not dependent on spatial context.

A user who is blind or has low vision can only read a small portion of the computer screen at a time. A user who has tunnel vision or uses a screen enlarger will see a control and perhaps its immediate surroundings, while a blind user examining a control will have only its name, its type, and the name of the window and any group box it is in. They will not have any of the context provided by spatial arrangements.

Labels do not have to be long and detailed. However, having several buttons with identical labels can be confusing if they are only distinguished by position. This confusion can be cleared somewhat if the position of the buttons in the tab order makes the association clear or if the buttons are within separately labeled group boxes.

The following illustration shows a dialog box where ambiguously named buttons are clarified by being placed in distinctively labeled group boxes and by having buttons immediately follow, in the tab order, the controls to which they are related.



Positioning Related Items near Each Other

You should try to arrange related items near each other. Because a person with low vision or tunnel vision can only see a portion of the screen at a time, this arrangement helps reduce the amount of work the person has to do to shift their gaze back and forth between related items. It also makes relationships clearer to all users.

Using Consistent and Expected Screen Layouts

People who use software to enlarge a portion of the screen or who use a screen review utility cannot see the entire screen at once. Placing screen components, such as buttons and toolbars, according to standard conventions can help these people get familiar with the product. It can also help screen reading software identify status items and other components that need special handling.

For more information about layout conventions, see *The Windows Interface Guidelines for Software Design.*

Spacing for a Specific Font

Some applications space their dialog boxes so tightly that they look unattractive if the user changes the dialog box font. Some users change the font to make the dialog box easier to read, so you should leave enough white space in your dialog box layouts that they can accommodate small changes in font metrics. Extra white space also makes an application easier to localize into other languages.

Some developers fear that a change of font will completely break their dialog boxes, but this is rarely a problem. Users typically change only the size of the font, not the typeface, and Windows automatically positions dialog box controls based on the size of the dialog box font.

The primary case where changing fonts can be a problem is when an application draws directly into elements of a dialog box. For example, some applications create a static control and then draw over it to create a custom design element. This element can appear incorrectly if the size of the static control is scaled to match a new dialog box font size. However, the application can determine the proper location and size at run time to avoid these problems.

Some applications now include specific fonts in their dialog boxes rather than relying on the system dialog box font. It is a dangerous practice, however, to not allow the user to adjust those font sizes. For more information about these issues, see "Color" and "Size" earlier in this document.

Optional Ease-of-Use Features

The following features are recommended because they make an application much easier to use for some users. However, their exclusion should not render an application inaccessible.

Providing Mouse Access to Common Features

Some users, who can use a pointing device but not a keyboard, use an on-screen keyboard utility to simulate keyboard input. Using the utility is less convenient and less efficient than performing actions designed for the mouse, so providing good mouse support makes the application easier for them—and others—to use.

Using Only Simple Mouse Operations

When possible, you should require only single-clicks to perform common operations. Some users have difficulty holding down a mouse button while moving the pointing device, and this makes drag-and-drop operations difficult. Double-clicking is also more difficult than single-clicking, so you should provide single-click access to commonly used operations whenever possible. You should also avoid requiring the use of mouse button 2, because some pointing devices and many alternative devices used by people with disabilities do not support it.

Reconfiguring Commands and Dialog Boxes

Providing customization of keyboard commands, menus, and dialog boxes makes it possible for users to specially tailor an application for their needs. For example, users who enter keystrokes slowly (either through the keyboard or other input devices) can benefit from assigning simple key combinations to lengthy, repetitive tasks. Users with cognitive disabilities can often benefit from a reduction in the number

of options in menus and toolbars. One approach is to provide the user with a choice of menus with different levels of complexity, such as novice, intermediate, and advanced menu configurations.

Macro capabilities that allow a user to create custom dialog boxes is also another good way to reduce complexity and improve access to the features a specific user requires. This type of feature is normally supported in large applications, such as Microsoft Word and Microsoft® Excel.

Making Graphical Decorations Optional

If your application uses graphical "decorations" that do not convey additional information, you should consider providing an option to hide them. For example, icons that illustrate option buttons can be hidden if the function of the buttons is already described by accompanying text. Such graphical decoration is very useful for most users, but it can be a hindrance for users with cognitive disabilities who require a simpler interface with less visual distraction.

Verification of an Application's Accessibility

All the planning in the world will not ensure that an application is as accessible as it could be. However, a number of techniques can be used to measure your success in meeting your accessibility objectives. These techniques are described in the sections that follow.

Testing for Compatibility with Accessibility Aids

Your application can be inaccessible if it has problems running with certain accessibility aids, such as screen review utilities, screen enlarger utilities, or voice input utilities. Even if you think of your product as an application, it is a development platform that accessibility aids must be compatible with.

If possible, you should try to include a sampling of utilities in any compatibility testing you perform. To obtain a list of vendors and their utilities, see Appendix A, "Additional Resources."

Including Accessibility Sites in Beta Tests

The best way to find out if your product is really usable by people with disabilities is to actively solicit their feedback. Include those people in any beta or usability tests you run on your product or employ them on your staff. You can also send evaluation copies to organizations that represent or work with people with disabilities.

You should also try to include companies who develop accessibility aids in your beta program. This will not only allow them to make sure their product works with your own, but also allow them to prepare any special configuration files or other items necessary to make the two products work well together. They can also provide you with valuable technical suggestions for improving your application's accessibility. For more information about accessibility aid manufacturers, see Appendix A, "Additional Resources."

Including Users with Disabilities in Usability Testing

If you perform usability testing, you can try to include subjects who have disabilities. You do not necessarily have to design special tests for these subjects, but you should watch to see how these individuals approach and perform the ordinary tasks you are already testing. It can be a very informative process, helping you learn about the different ways in which people work.

Comparing Against the Accessibility Guidelines

You should assign testing resources that compare your product against the guidelines described in this document.

Try It Out!

You can test your application to see how well it addresses some of the issues discussed in this document by following these suggestions:

- Use your computer for a week after choosing a high contrast color scheme in Control Panel, especially one that uses white text on a black background, such as High Contrast Black. Are there any portions of your application that become invisible or difficult to use or recognize?

- Use your computer for a week without a mouse. Drop the mouse down behind your desk. Are there operations you cannot perform? Is anything especially awkward to use? Are the keyboard mechanisms adequately documented?
- Increase the size of your system font using the Display property sheet. (In Windows version 3.1, edit the Fonts.fon=, Oemfonts.fon=, and Fixedfon.fon= entries in the SYSTEM.INI file to specify a larger font. It is easy to test with the 8514*.fon files provided with Windows.) Does your application look good despite the changes? Can you adjust all of the fonts in your application to be at least as large as the system font?
- Change the display scaling ratio using the Custom Font Size feature in the Display property sheet. Does your application appear consistent, or do various elements of the user interface appear disproportionately large or small?
- Use your computer for a week after choosing an enlarged appearance scheme in Control Panel, such as Windows Standard (Extra Large). Are there any portions of your application that are not consistent with other applications?

Appendix A: Additional Resources

General Resources

For more information about Microsoft products and services for people with disabilities, or to obtain a listing of third-party accessibility aids for Windows and Windows NT or a listing of resources available to help produce accessible documentation, contact:

Microsoft Sales Information	Voice telephone	(800) 426-9400
Center One Microsoft Way	Text telephone	(800) 892-5234
Redmond, WA 98052-6393	Fax	(206) 635-6100

The Trace Research and Development Center of the University of Wisconsin at Madison produces a book and a compact disc describing products that help people with disabilities use computers. The book, titled *Trace Resource Book,* provides descriptions and photographs of about 2,000 products. The compact disc, titled *CO-NET CD*, provides a database of more than 18,000 products and other information for people with disabilities. It is issued twice a year. To obtain these directories, you should contact:

Trace R&D Center	Voice telephone	(608) 263-2309
S-151 Waisman Center	Text telephone	(608) 263-5408
1500 Highland Avenue	Fax	(608) 262-8848
Madison, WI 53705-2280		

For general information and recommendations about how computers can help specific users, you should consult a trained evaluator who can best match the user needs with the available solutions. An assistive technology program in your area will provide referrals to programs and services that are available to you. To locate the assistive technology program nearest you, you should contact:

National Information System	Voice/tex	(803) 777-4435
Center for Developmental Disabilities	telephone	(803) 777-6058
Benson Building	Fax	
University of South Carolina		
Columbia, SC 29208		

Additional Accessibility Guidelines

This document is based on the general guidelines first proposed in the white paper "Making Software More Accessible for People with Disabilities." That paper was prepared by Gregg Vanderheiden of the Trace R&D Center of the University of Wisconsin at Madison under funding from the Information Technology Foundation (formerly ADAPSO Foundation) and the National Institute for Disability and Rehabilitation Research (NIDRR) of the U.S. Department of Education. That paper and similar guidelines for other types of products are available on the *CO-NET CD* or in print from:

Trace R&D Center	Voice telephone	(608) 263-2309
S-151 Waisman Center	Text telephone	(608) 263-5408
1500 Highland Avenue	Fax	(608) 262-8848
Madison, WI 53705-2280		

The Windows Interface Guidelines for Software Design, included in the Win32 SDK and available as a book from Microsoft Press, contains a section on accessible software design.

The Windows Interface: An Application Style Guide, published by Microsoft Corporation and included with the Microsoft Windows SDK version 3.1, contains useful information on user-interface design for Windows version 3.1.

Writing Accessible HTML Documents, a guide to creating accessible documents for the World Wide Web, is available from the Center for Information Technology Accommodation, General Services Administration, Washington D.C.

Customizing for a Specific Operating System

There are many ways you can customize the appearance and behavior of Windows or Microsoft® Windows NT™ to accommodate varying vision and motor abilities without requiring any additional software or hardware. These customizations include adjusting the operating system's appearance, as well as the behavior of the mouse and keyboard. The specific methods available depend on which operating system you are using. Application notes are available describing the specific methods available for each operating system.

The Microsoft Windows Guidelines for Accessible Software Design
62
For information related to customizing your operating system for people with disabilities, see the appropriate application note for the following operating systems.

Operating system	Application note
Microsoft Windows, version 3.0	WW0786.TXT
Microsoft Windows, version 3.1	WW0787.TXT
Microsoft Windows for Workgroups, version 3.1	WG0788.TXT
Microsoft Windows NT, versions 3.1 and 3.5	WN0789.EXE
Microsoft Windows 95	WN1062.EXE

These application notes are available for downloading from the following network services:

- MSN™ the Microsoft Network online service.
- CompuServe®.
- GEnie™.
- Microsoft Download Service (MSDL), which you can reach by calling (206) 936-6735 any time, except between 1:00 A.M. and 2:30 A.M. Pacific time. To reach the number use the following communications settings:

Setting	Value
Baud rate	1200, 2400, 9600, or 14400
Parity	None
Data bits	8
Stop bits	1

- Various user-group bulletin boards (such as the bulletin-board services on the Association of PC User Groups network).
- In /SOFTLIB/MSLFILES on the Internet servers FTP.MICROSOFT.COM, GOPHER.MICROSOFT.COM, and WWW.MICROSOFT.COM.

People within the United States who do not have a modem can order these application notes on disks by calling the Microsoft Sales Information Center at (800) 426-9400 (voice telephone) or (800) 892-

5234 (text telephone). In Canada, you can call (905) 568-3503 (voice telephone) or (905) 568-9641 (text telephone).

Note

Customers outside the United States can contact the Microsoft subsidiary in their country to find out about the availability of application notes and other resources in their area.

Appendix B: Documentation, Packaging, and Support

Most of this document describes the process of designing and building software, but accessibility should also be considered in the process of producing, marketing, and supporting software products.

Providing Documentation in Alternative Formats

Some users have difficulty reading or holding conventionally printed documentation, so documentation should also be provided in other more accessible formats, such as online versions.

You should inform the user if he or she has or can obtain online documentation that includes all or almost all of the information in the printed versions. It is also acceptable to have complete online documentation included only with a CD-ROM version of your product rather than on floppy disks. In either case, the user should be able to easily determine if online documentation is available, how complete it is, and how to obtain it. Of course, it is also necessary to make sure that the application presenting the documentation is itself accessible.

Software vendors should allow customers to order the documentation on floppy disk. This type of electronic documentation is normally provided as formatted ASCII text files, and this format addresses a wide variety of needs. For example, customers who are blind or have low vision can read the files in their own word processor using screen review or screen enlarger utilities, and customers with mobility impairments can read them online without holding or turning the pages of a physical book. The ASCII files normally include special tags that identify the structure of the document (that is, tags for headings, footnotes, and so on).

Documentation can also be provided in alternative formats, such as large print, Braille, or audio tapes. Most companies do not provide documentation in these formats, but license instead the source files for documents to users or organizations who want to create accessible versions in those formats.

A list of resources who can provide additional information or can help in translating or distributing your documentation in accessible format can be found in Appendix A, "Additional Resources."

Conveying Information with Text and Graphics

In general, accessible documentation design follows the same rules as accessible visual design for software:

- Information should not be conveyed by color or graphics alone. If printed documentation relies on color or graphics to convey important information, that information might not be available to some customers. Some customers may rely on a variety of devices to enlarge a document or translate it into ASCII text, speech, or Braille, and those devices are often unable to preserve graphic or color information.
- Color and graphics should be added redundantly to the text to improve documents. For example, if a reference work contains a list of function calls and gives important information about each one, some entries can be printed in blue ink, rather than black, to make it instantly obvious that they are not supported on all systems. In this case, all the entries that are shown in blue can also include a phrase, such as "platform specific" in their description. If space is limited, each can simply be marked with an asterisk. Such redundant use of information often makes documentation easier for everyone to use. A phrase or asterisk could also be used in cases where certain paragraphs are called out with a graphic in the margin.

  Modifying text in this way makes it easier for you, or another organization working on your behalf, to translate your documentation into alternative formats, such as Braille or online documentation.
- Maintain high contrast between text and its background, and avoid screened art behind text.
- Text should not be less than 10 points in size.

Making Diskettes Easily Identifiable

All diskettes and CD-ROM disks should be given a unique volume label that easily identifies the specific product and disk number. People who are blind may not be able to read the printed disk label, but providing an appropriate volume label allows them to identify the diskette using the **dir** command at the command prompt.

Making Packaging Easy to Open

Users with mobility impairments may have trouble opening some packages. It can be useful to examine packaging to see if it could be made easier to use. For example, shrink-wrapped packages can be easy to open if they are left unsealed at a place where two layers overlap.

Providing Customer Support through Text Telephone and Modem

Customers who are deaf or hard-of-hearing or who have speech impairments may not be able to use standard voice telephones to access customer information and support services. These services should, therefore, be made available through a text telephone (also known as TT or TDD) and standard ASCII modems. Stand-alone text telephones are available with a wide range of features, and combination TT/ASCII modems can also be attached to standard computers, although specialized software is normally used to get full answering-machine functionality. For additional information, including lists of vendors supplying text telephone hardware and software, see Appendix A, "Additional Resources."

Binding Documents to Lie Flat

Printed documentation can be bound in a large number of different ways, but comb and spiral bindings are generally considered the most accessible because they allow a document to lie flat. These types of bindings are useful for people with motion or visual impairments; for example, a person who is quadriplegic may lie the book flat and turn the pages with a pencil, a person who is blind may run it through a flatbed scanner to use optical-character recognition for conversion to an online format, and a person with low vision might use a closed-caption television system to enlarge the pages. Flat bindings are also preferred by people who want to be able to type while reading.

Generally, the choice of binding is made on a purely economic basis. Often a method such as perfect (glue) binding is considerably cheaper than other alternatives, but it is not particularly convenient for users with disabilities.

Appendix C: Windows Version 3.*x* Guidelines

The following guidelines do not affect Win32®–based applications designed for Windows 95 or Windows NT, but should be followed when developing 16-bit applications (also called Win16 applications) for Windows version 3.*x*.

Yielding Control to Background Applications

Windows-based 16-bit applications should yield control at all times so that other programs, such as accessibility aids, can run in the background. If a program refuses to yield control, the user is unable to access the machine. You can avoid access problems by following these techniques:

- Avoid using system modal dialog boxes or windows. When a system-modal window is active, no background tasks are allowed to run. (This is true to a lesser extent for 16-bit applications running under Windows 95, although it is not a problem for those running under Windows NT.)
- Avoid using the **PeekMessage** function in tight loops without yielding.

Colors in Online Help

An author that is designing an online help topic can specify foreground and background colors, or use the color scheme selected by the user in Control Panel. If the author specifies his or her own color scheme, the user running Windows version 3.*x* or Windows NT version 3.*x* has no way to override the scheme and use a different set of colors. As a consequence, some users who require high-contrast colors schemes will not be able to make use of the help topic. (In Windows 95, the help system provides the user with the option to use only Control Panel colors. However, this option leads to having the topic appear different from the help author's preference.)

If you want your online help to be usable by as many customers as possible, you should generally allow the user to choose their own color scheme rather than specifying one of your own choosing.

Testing Accessibility Flags

Windows 95 provides four new flags that advise applications when they should adjust their behavior to accommodate users with disabilities. Although each can be tested using the **SystemParametersInfo** function, this capability is not supported in earlier versions of Windows or Windows NT. To make this behavior available in earlier operating systems, you can test for the flags as WIN.INI settings. These settings are not used by Windows itself, but they can be set manually by users who want the specified disability options. The following WIN.INI settings are recommended.

SystemParametersInfo	WIN.INI setting in earlier operating systems
SPI_SHOWSOUNDS	[Windows] ShowSounds=TRUE
SPI_KEYBOARDPREF	[Windows] KeyboardPref=TRUE
SPI_SCREENREADER	[Windows] ScreenReader=TRUE
SPI_HIGHCONTRAST	[Windows] HighContrast=TRUE

Designed for Microsoft® Windows NT® and Windows® 95

Logo Handbook for Software Applications

Version 2.0
July, 1996